Enhancing e-Infrastructures with Advanced Technical Computing: Parallel MATLAB on the Grid

Chakravarti, A.J.
(The MathWorks, Inc. Natick, MA, USA) *et al*

13 May 2008

EGEE-II is a project funded by the European Commission
Contract number INFSO-RI-031688

The electronic version of this EGEE Technical Reports is available
on the CERN Document Server at the following URL:
<http://cdsweb.cern.ch/search.py?p=EGEE-TR-2008-001>

Enhancing e-Infrastructures with Advanced Technical Computing: Parallel MATLAB® on the Grid

Arjav J. Chakravarti, Silvina Grad-Freilich
The MathWorks, Inc. Natick, MA, USA

Erwin Laure
CERN, Geneva, Switzerland

Michel Jouvin, Guillaume Philippon, Charles Loomis
Laboratoire de l'Accélérator Linéaire (LAL), Université Paris-Sud, Orsay, France

Evangelos Floros
GRNET S.A., Athens, Greece

Abstract

MATLAB® is widely used within the engineering and scientific fields as the language and environment for technical computing, while collaborative Grid computing on e-Infrastructures is used by scientific communities to deliver a faster time to solution. MATLAB allows users to express parallelism in their applications, and then execute code on multiprocessor environments such as large-scale e-Infrastructures. This paper demonstrates the integration of MATLAB and Grid technology with a representative implementation that uses gLite middleware to run parallel programs. Experimental results highlight the increases in productivity and performance that users obtain with MATLAB parallel computing on Grids.

I. Introduction

a) MATLAB and Parallel Computing

MATLAB [11] is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. MATLAB is used to solve problems in several application areas such as signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. Add-on toolboxes extend the MATLAB environment to solve particular classes of problems in different application areas.

Simulink® [12] is a companion product to MATLAB that offers an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that allows users to design, simulate, implement, and test a variety of time-varying systems.

MATLAB, Simulink and add-on products provide engineers, scientists, mathematicians, and educators with a powerful set of tools. These tools serve a broad range of tasks across a variety of industries from automotive and electronics to finance and telecommunications. There are an estimated 1 million MATLAB users in industry and academia worldwide. MATLAB has been especially widely adopted by the academic field. More than 3,500 universities across the world use MATLAB for research and teaching.

Engineers and scientists are solving increasingly complex problems, with running times and data sets that far exceed the capabilities of traditional uniprocessor systems. Simultaneously, advances in computer processing power have enabled easy access to multicore processors as well as clusters built from commercial-off-the-shelf (COTS) components. These two factors have driven the demand that MATLAB easily exploit multiprocessor architectures.

The MathWorks responded to this demand by beginning to support parallel computing in MATLAB. Two products were released in 2004: Distributed Computing Toolbox (since renamed to Parallel Computing Toolbox) and MATLAB® Distributed Computing Engine (since renamed to MATLAB® Distributed Computing Server) [13,14]. Since then, MATLAB and Simulink users have taken advantage of parallel programming and execution on multicore desktop computers as well as personal, workgroup, departmental and enterprise clusters [16,17,18,19,20]. Section II contains a detailed description of parallel MATLAB technology and products.

b) Grid Computing and EGEE

Modern science is increasingly dependent on ICT technologies, analysing huge amounts of data (in the TeraByte and PetaByte range), running large scale simulations requiring thousands of CPUs, and sharing results between different research groups. This collaborative way of doing science has led to the creation of *Virtual Organizations (VOs)* that combine researches and resources (instruments, computing, data) across traditional administrative and organizational domains [1]. Advances in networking and distributed computing techniques have enabled the establishment of such VOs and more and more scientific disciplines are leveraging VOs to do *Grid* computing [2,3,4].

The past years have shown the benefit of basing Grid computing on a well managed infrastructure federating the network, storage, and compute resources across different institutions and making them available to different scientific communities via well defined protocols and interfaces exposed by a software layer (*Grid middleware*). A number of Grid infrastructures have been established in the past years like EGEE [23] and DEISA [24] in Europe, OSG [25] and TeraGrid [26] in the US, and APAC [27] and NAREGI [28] in the Asia-Pacific.

Among these projects, the EGEE (Enabling Grids for E-sciencE) project unites thematic, national and regional Grid initiatives in order to provide an e-Infrastructure available to

all scientific research in Europe in support of the European Research Area. EGEE has also expanded to the Americas and Asia Pacific working towards a world-wide e-Infrastructure. The project is a multi-phase programme starting in 2004 and expected to end in 2010, when the federations will govern the use of the grid and its technology. EGEE currently federates some 250 resource centres from 48 countries providing over 50,000 CPUs and several PetaBytes of storage.

The EGEE infrastructure is being used by over 5000 users forming some 200 VOs and running over 140,000 jobs per day. EGEE users come from disciplines as diverse as archaeology, astronomy, astrophysics, computational chemistry, earth science, finance, fusion, geophysics, high energy physics, life sciences, material sciences, and many more. EGEE also works with industrial users and industrial service providers to ensure technology transfer to business. [10] provides further details on EGEE's work with business.

c) Parallel MATLAB on the Grid

MATLAB and Grid computing are recognized as key enablers of engineering and scientific activity. The integration of these technologies is a natural demand of the scientific and industrial community: MATLAB users will be able to solve larger problems by running their applications in parallel on collaborative Grids, while Grid users will have the ability to develop complex applications in MATLAB.

A requirement for the integration of MATLAB and Grid computing is that a user be able to develop a MATLAB application without needing to consider the variety of environments within which it might be executed. This paper presents a solution that gives a user that capability. The contributions of this paper are

i) Description of the MATLAB parallel programming paradigm in which the language is independent of the execution environment.
ii) Implementation of a mechanism for the execution of parallel MATLAB programs on Grids through gLite middleware.
iii) Experimental results from the execution of a MATLAB application on an EGEE Grid.

II. Parallel MATLAB

The MATLAB language is well suited to rapid prototyping and development of technical computing applications. MATLAB offers the ability to express ideas in a language close to mathematical expression, within an interactive development environment. Multiplatform support and add-on toolboxes make MATLAB programs portable, and easy to write, run and maintain.

Parallel language design for MATLAB draws from Lurie's annotation-based language model [15], in which domain experts make minimal annotations to their high-level code to express an intention of using multiple compute resources. Parallelism has been built into the MATLAB language through a set of functional constructs and data structures. The language is independent of execution environment and resource allocation, which enables a user to program a parallel application without making changes to existing code or workflows.

a) Programming and Execution

MATLAB and Parallel Computing Toolbox (PCT) provide the development and execution environments for parallel applications on a user's workstation. PCT can launch up to 4 MATLAB *worker* processes (MATLAB computational engines that run independently of client sessions) on the workstation, which enables the user to locally test and debug a parallel program, as well as to utilize multiple processing cores. MATLAB Distributed Computing Server (MDCS) provides the execution environment for applications on cluster or Grid computers. Several MATLAB workers can be used to execute applications in parallel.

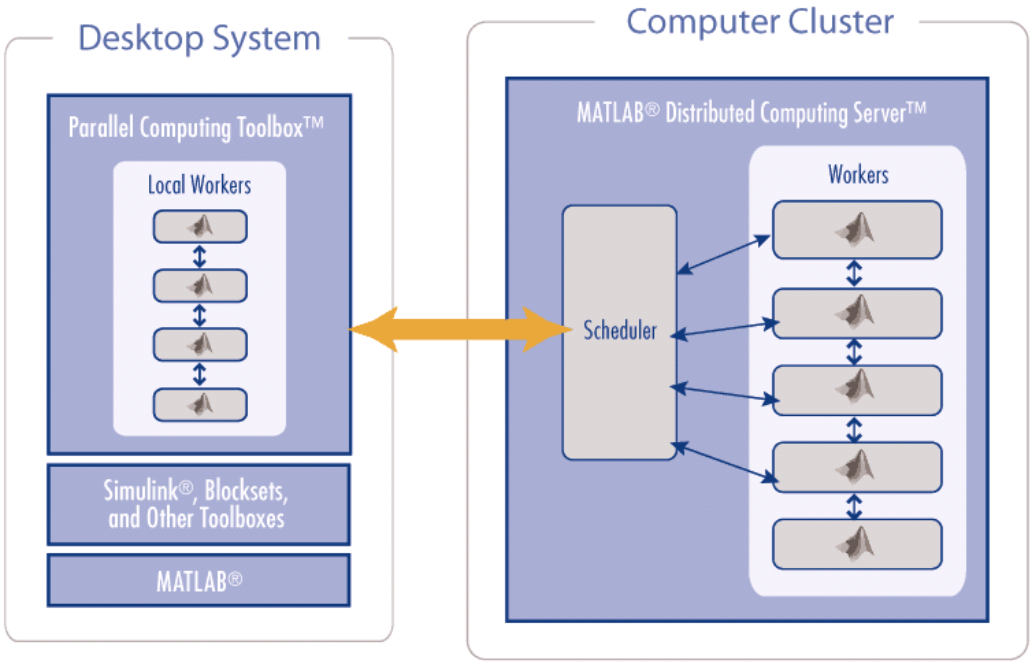

Figure 1: Parallel Programming and Execution

b) Parallel Language

MATLAB provides a set of constructs that can be applied to exploit various types of parallelism with minimal effort. Users can choose a specific subset of these constructs in order to exploit parallelism in their applications. Most MATLAB users will choose high-

level constructs like parallel for loops and distributed arrays, while advanced parallel programmers might use low-level message passing functions.

The parallel for loop, *parfor,* exploits task parallelism. A user expresses that the iterates of a *for* loop are order-independent by annotating the *for* as a *parfor*. During program execution, the *parfor* iterates are distributed to be run on available MATLAB workers, potentially across multiple physically-separated computers. If no MATLAB workers are present, for instance on a single-processor system, *parfor* behaves like a traditional *for* loop.

```
% Original for loop          % Parallel for loop
for itr = m : n              parfor itr = m : n
    % loop body                 % loop body
End                          End
```

Figure 2: Parallel for loop

MATLAB distributed arrays target data parallelism by implementing the Partitioned Global Address Space (PGAS) model across MATLAB workers. In the PGAS model for SPMD (Single Program Multiple Data) programs, multiple SPMD threads or processes share a part of their address space. By using MATLAB distributed arrays, users can allocate matrices across the available MATLAB workers and work with very large data sets. MATLAB operations like matrix multiplication, decomposition, and transforms also work directly on distributed arrays. Some of these operations leverage ScaLAPACK.

An increasing number of add-on products to MATLAB, such as Optimization Toolbox™ [29], Genetic Algorithm and Direct Search Toolbox™ [30], and SystemTest® [31], contain built-in support for parallel computing. Users of these tools can take advantage of multiprocessor resources *without* writing a single line of parallel code.

Parallel computing experts have the option of using low-level functionality to exercise greater control over their applications: batch jobs that contain dependent or independent tasks; MATLAB message passing functions that are wrappers around commonly-used MPI (MPICH2) operations.

c) Interactive Development

Interactivity makes MATLAB a uniquely simple environment to understand and use. Prototyping and algorithm development occur in a simple, iterative fashion from a command line, leading to the creation of an application in the form of MATLAB files. This experience is also available to developers of parallel code. It is particularly useful for programmers to test and debug programs on their workstations before submitting any batch jobs. *matlabpool* command and *Parallel Command Window* allow a user to open an interactive session to a selection of MATLAB workers. The user can test and profile code, explore distributed data and detect deadlocks. Batch jobs can be submitted only after the user is confident of the correctness and performance of the parallel application.

d) Selection of Execution Environment

The separation of language from execution environment allows a parallel MATLAB program to run correctly and scale up to whatever computing resources are made available to it. The location and nature of the computing resources are saved in a *Distributed Configuration*. A user or administrator can create Configurations for a variety of systems, e.g., a multicore workstation, a workgroup cluster, or a large-scale Grid system. By selecting one of the Configurations, the user defines the resources on which the parallel MATLAB program will be executed, without modifying any code.

III. EGEE

The computing and storage resources EGEE integrates are provided by a large and growing number of Resource Centres (RCs), mostly in Europe but also in the Americas and Asia Pacific. The EGEE infrastructure is federating resources and making them easily accessible but does not own the resources itself. Instead, the resources accessible belong to independent resource centres that procure their resources and allow access to them based on their particular funding schemes and policies. Federating the resources through EGEE allows the resource centres to offer seamless, homogenous access mechanisms to their users as well as to support a variety of application domains through the EGEE VOs. Hence, EGEE on its own cannot take any decision on how to assign resources to VOs and applications. EGEE merely provides a market place where resource providers and potential users negotiate the terms of usage on their own. EGEE provides a continuous service to its users through its RCs that are managed via *Regional Operations Centres (ROCs*) taking over the responsibility of managing the RCs in their region. Regions are defined geographically and include up to 8 countries. This setup allows adjusting the operational procedures to local peculiarities like legal constraints, languages, best practices etc. In addition, the *Global Grid User Support* (GGUS) system is used throughout the infrastructure as central entry point for managing problem reports and tickets, for operations, as well as for user, VO, and application support.

EGEE deploys the gLite middleware [5], a middleware distribution that combines components developed in various related projects, in particular Condor [6], the Globus Toolkit (GT) [7], LCG [8], and VDT [9], complemented by EGEE developed services. This middleware provides the user with high level services for scheduling and running computational jobs, accessing and moving data, and obtaining information on the Grid infrastructure as well as Grid applications, all embedded into a consistent security framework.

EGEE actively engages with application communities, beginning with High Energy Physics (HEP) and Biomedicine at the start of EGEE but expanding to support many other domains. These include Astronomy, Computational Chemistry, Earth Sciences, Financial simulations, Fusion science and Geophysics, with many other applications

currently evaluating the infrastructure. EGEE also works with industrial users and industrial service providers to ensure technology transfer to business. Further details on EGEE's work with business can be found at [10].

The applications running on the EGEE Grid are rapidly moving from testing to routine usage, with some communities already running large numbers of jobs on a daily basis. Overall, the infrastructure serves some 140,000 jobs per day. The figure below shows the development of the usage of the infrastructure, normalized to kilo SpecInt200. There has been a steady increase in usage and although the High Energy Physics domain is still the dominant user responsible for some 2/3 of the resource consumption, the usage by other domains is steadily increasing and now equivalent to the total HEP usage a year ago. EGEE allows user groups to federate their distributed resources into a seamlessly accessible infrastructure thus optimizing their usage; at the same time, resource centres supporting multiple disciplines can offer their services to all the supported disciplines via a common interface provided by EGEE. [3] provides an overview on applications using EGEE.

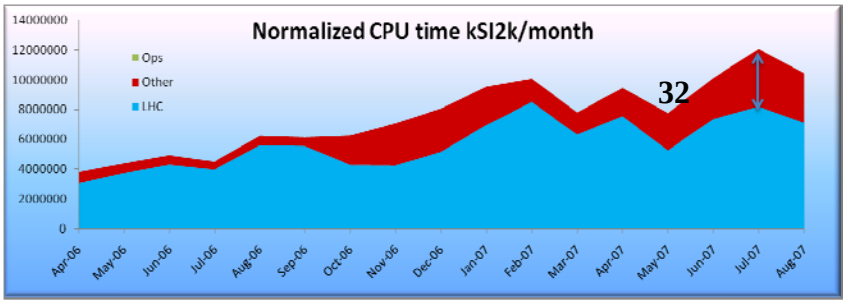

Figure 3: Usage of the EGEE resources

IV. Parallel Execution of MATLAB on the Grid

A MATLAB user parallelizes an application by using high-level language constructs like *parfor* or low-level message-passing functions. The user does not need to write any special code to account for the fact that the application will be run on a Grid. Executing the application on a Grid system typically involves the following steps.

i) The user selects a "local" Distributed Configuration and tests the parallel program with local MATLAB workers on a workstation.

ii) Once the program runs successfully, the user selects a Configuration for the Grid system and runs the program again. This time, with no intervention or direction from the user, MATLAB will transfer input data and code to the Grid, execute the parallel code on MATLAB workers, and retrieve output data.

a) MATLAB integration with gLite

gLite middleware provides the user with high level services for scheduling and running computational jobs. In order to launch and control gLite jobs, a user requires a remote

command execution and file copying mechanism between workstation and Grid. This can be done by installing gLite client tools, using https, or setting up passwordless ssh between workstation and remote Grid UI node.

When a user runs a parallel application, MATLAB *automatically* generates Job Definition Language files and other gLite-specific scripts, copies input files to a Storage Element (SE) and registers the files with a File Catalog. Submission commands are then called to submit gLite jobs. The user can continue to use MATLAB to do other work while the jobs run on the Grid.

The Resource Broker of the gLite middleware locates sites that have MDCS installations. MATLAB workers are started up on Worker Nodes and the application is executed in parallel. Output data is written to the SE.

The MATLAB session on the user's workstation automatically runs a timer that periodically check the state of the gLite jobs. When a gLite job finishes, its output data and files are copied from the SE to the workstation. The user then retrieves the output of the parallel MATLAB application and destroys the job. The destruction of the MATLAB job includes the deletion of files that are no longer needed on the SE.

V. Experiments

a) Parameter Sweep Application

A MATLAB application from the field of systems biology was chosen for experimental runs. The application was created using SimBiology® [21], which extends MATLAB® with tools for modeling, simulating, and analyzing biochemical pathways.

The application allows a user to hypothesize and simulate the effects of drug interactions with a model of Caspase-induced apoptosis (programmed cell death) [22]. A parameter sweep is performed across a range of drug-injection rates to quantify the effect of the drug dosage on the Caspase3 activation levels. Several hundred thousand simulations might need to be performed, requiring several days of computation on one MATLAB.

b) Experimental Setup

MATLAB and PCT were installed on a dual-core 1.83 GHz T60 Windows laptop with 2GB RAM. The laptop was located at The MathWorks headquarters in Natick, USA.

MDCS was installed on a cluster at Laboratoire de l'Accélérator Linéaire (LAL) in Orsay, France. The cluster contained 33 IBM rack-mounted servers, out of which 8 were used for the experiments. Every server had 2 CPUs (Intel Woodcrest 2.3 GHz), each with 4 cores, as well as 16 GB of RAM and a 160 GB hard disk. The batch system used on the

cluster was Torque 2.1.9, with a Maui 3.2.6 scheduler. A standard gLite 3.1 configuration was used for the cluster. One of the rack-mounted servers acted as the Grid interface node.

Remote command execution and data transfer between the workstation and the Grid interface node was done using SSH and a PuTTY client. Once passwordless SSH login was established, MATLAB would call *plink* to execute remote commands and *pscp* to transfer data.

c) Measurements

The size of the problem being solved by the chosen parallel MATLAB application was determined by the number of simulations that it ran. The performance of the application on the EGEE Grid was measured by varying the problem size with the number of MATLAB workers such that the load per worker was maintained as a constant.

The total running time of the application was measured on the user's workstation. Therefore, in addition to the time spent on MATLAB simulations, this value also included the overhead of job submissions to gLite, of data transfers between workstation and Grid, and of the Resource Broker scheduling the application on Grid resources. As presented in Table 1 below, the application was run on different numbers of workers while maintaining the number of simulations executed by each worker at 11500. Speedup ratio is the ratio of the running time of 11500 simulations on one worker to the running time of 11500* *p* simulations on *p* workers.

Number of Simulations (Problem Size)	Number of MATLAB workers	Running Time in seconds	Simulations per second	Speedup Ratio
11500	1	3854	2.98	1.00
23000	2	3885	5.92	0.99
46000	4	4000	11.50	0.96
92000	8	4207	21.87	0.92
184000	16	4428	41.55	0.87
368000	32	4974	73.98	0.77
736000	64	5960	123.49	0.65

Table 1: Parallel Performance for Scaled Problem Size

Figure 4 shows that the rate at which simulations are performed by the MATLAB application increases with the number of MATLAB workers that are executing the simulations in parallel. The figure also indicates that the overhead to distributing an application increases with an increase in the number of resources over which the distribution is performed. A significant amount of the overhead is due to the fact that data is transferred between remote sites in the US and France for each job that is submitted to gLite.

The experimental results illustrate that a MATLAB user can obtain very significant increases in productivity by running parallel programs on e-Infrastructures. 64 MATLAB workers take 100 minutes to run 736000 simulations while a single MATLAB would take close to 3 days to solve the same problem. Parallel MATLAB on the Grid directly accelerates the pace of engineering and scientific endeavor by allowing users to rapidly test, iterate and arrive at solutions.

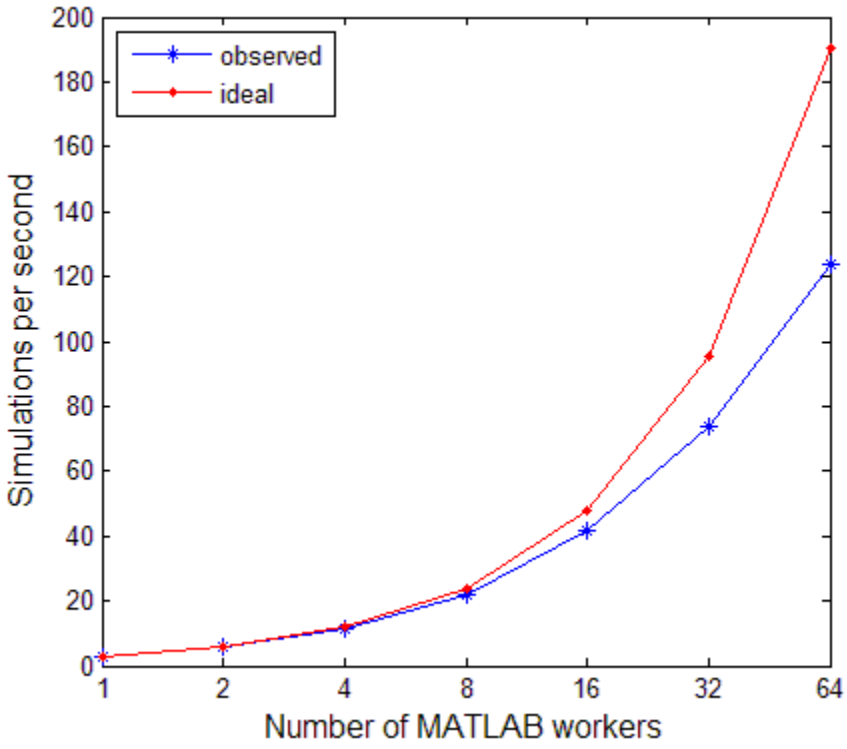

Figure 4: Simulations per second for Scaled Problem Size

VI. Conclusion

This paper describes the integration of MATLAB and Grid technology, and the use of large-scale e-Infrastructures like EGEE to run parallel MATLAB applications. Language features and constructs within MATLAB enable users to interactively develop parallel programs, either by annotating existing sequential code or by authoring new parallel code. The programs are independent of their execution environment which makes them well-suited to be run within a Grid system where heterogeneous computational resources are allocated based on current availability.

The mechanisms of MATLAB integration with gLite Grid middleware have been implemented and explained in this paper. Experimental results from running a MATLAB application on an EGEE system have also been presented. The results illustrate that users can substantially improve their productivity by running parallel MATLAB programs on

Grids. The integration of these two technologies brings the benefits of high-productivity computing to a large community of engineers and scientists.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid", The International Journal of High Performance Computing Applications, 15, 200-222 (2001).
[2] I. Foster, C. Kesselman (Eds.), "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 2003.
[3] V. Floros et al. (Eds.), "EGEE User Forum – Book of Abstracts", Technical Report EGEE-TR-2007-002, http://cdsweb.cern.ch/record/1068842, November, 2007.
[4] M. Lamanna and E. Laure, editors. "The EGEE User Forum - Experiences in using Grid Infrastructures", volume 6(1) of Journal of Grid Computing. Springer, March, 2008.
[5] E. Laure, S. Fisher, A. Frohner, et al.. "Programming the Grid with gLite" in, *Computational Methods in Science and Technology*, 12(1):33-45, 2006.
[6] D. Thain, T.Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience" *Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005*
[7] I. Foster. "Globus Toolkit Version 4: Software for Service-Oriented Systems". *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2-13, 2005.
[8] For details see http://lcg.web.cern.ch/LCG/
[9] For details see http://vdt.cs.wisc.edu/
[10] S. Holsinger, "Annual Report of the Industry Forum", EGEE-II Deliverable DNA2.5.1, https://edms.cern.ch/document/819325/, April, 2007.
[11] "Getting Started with MATLAB®", The MathWorks, Natick, MA, March 2008.
[12] "Using Simulink®", The MathWorks, Natick, MA, March 2008.
[13] "Parallel Computing Toolbox User's Guide", The MathWorks, Natick, MA, March 2008.
[14] "MATLAB® Distributed Computing Server System Administrator's Guide", The MathWorks, Natick, MA, March 2008.
[15] Roy Lurie, "Language Design for an Uncertain Hardware Future", HPCwire, September 2007.
[16] Silvina Grad-Freilich, "MATLAB embraces HPC", Desktop Engineering Magazine, February 2006.
[17] Cleve Moler, "Parallel MATLAB: Multiple Processors and Multiple Cores, The MathWorks News and Notes, June 2007.
[18] Glen White, "Controlling Electron Beams in the International Linear Collider", The MathWorks News and Notes, October 2006.
[19] Narfi Stefansson, Kelly Luetkemeyer and Rob Comer, "Accelerating a Geospatial Application using MATLAB® Distributed Computing Tools", The MathWorks News and Notes, January 2006.
[20] Jason R. Ghidella, Amory Wakefield, Silvina Grad-Freilich, Jon Friedman and Vinod Cherian, "The Use of Computing Clusters and Automatic Code Generation to Speed Up Simulation Tasks", AIAA Modeling and Simulation Technologies Conference and Exhibit, Hilton Head, SC, August 2007.
[21] "SimBiology® User's Guide", The MathWorks, Natick, MA, October 2007.
[22] BB Aldridge, G Haller, PK Sorger and DA Lauffenburger, "Direct Lyapunov exponent analysis enables parametric study of transient signalling governing cell behaviour", Syst Biol (Stevenage). 2006 Nov;153(6):425-32
[23] For details see http://www.eu-egee.org
[24] For details see http://www.deisa.org
[25] For details see http://www.opensciencegrid.org
[26] For details see http://www.teragrid.org
[27] For details see http://grid.apac.edu.au/
[28] For details see http://www.naregi.org
[29] "Optimization Toolbox User's Guide", The MathWorks, Natick, MA, March 2008.
[30] "Genetic Algorithm and Direct Search User's Guide", The MathWorks, Natick, MA, March 2008.
[31] "SystemTest User's Guide", The MathWorks, Natick, MA, March 2008.